TABLE OF CONTENTS

Unicom Tic Management System

Project Submission Report

---

1. Project Overview (Point-Wise)

Key Futures Implemented:

- Login System with role-based access for Admin, Staff, Students, and Lecturers.
- Course and Subject Management module.
- Student Management with course association.
- Exam and Marks Management with entry and viewing privileges by role.
- Timetable Management including Computer Lab and Lecture Hall allocation.
- SQLite database with relationships between tables.
- Role-based dashboards that restrict access to appropriate features.
- Error messages and input validation included.
- Simple and intuitive WinForms user interface.

Technologies Used:

- Programming Language: C#
- Framework: WinForms (.NET Framework)
- Database: SQLite using System.Data.SQLite
- IDE: Visual Studio

- UI Elements: Buttons, ComboBoxes, DataGridViews, TextBoxes, Labels,DateTimePicker

Concepts Used:

- OOP (Object-Oriented Programming)
- Encapsulation
- Parent and Child Classes
- MVC (Model–View–Controller) Architecture
- Role-Based Access Control
- Constructure OverLoading

Challenges Faced and Solutions:

- Role-Based Dashboard Visibility: Solved by checking user role after login and showing/hiding buttons accordingly.
- Room Allocation with Combo Box: Created a separate table for Rooms with room type filtering, and populated Combo Box dynamically.
- Login Validation: Added a simple user validation system querying the Users table and managing sessions based on roles.
- Database Table Creation: Implemented DatabaseManager.cs to auto-create tables if they don’t exist.
- Navigation Between Forms: Used controller logic to open/close forms based on user role without error.

2. Code Samples (Screenshots)

- Load Marks to MarksView Based On Role And Search

```
5 references
internal class MarkController
{
    1 reference
    public static void AddStudentExam(Exams exam)
    {
        using(SQLiteConnection connect = DatabaseManager.GetConnection())
        {
            SQLiteCommand cmd =connect.CreateCommand();
            cmd.CommandText = @"INSERT INTO ExamMarks (ExamsID, StudentsID)SELECT
                                @id, StudentsID FROM StudentSubject WHERE SubjectsID = @sid;";
            cmd.Parameters.AddWithValue("@id",exam.ID);
            cmd.Parameters.AddWithValue("@sid",exam.SubjectID);
            cmd.ExecuteNonQuery();
        }
    }
    1 reference
    public List<Marks> ViewExamMArks(string role, int UserID,string Search)
    {
        int StudentID = 0;
        if (role == "Student")
        {
            StudentID = StudentController.GetStudentID(UserID);
        }
        List<Marks> list = new List<Marks>();
        using (SQLiteConnection connect = DatabaseManager.GetConnection())
        {
            SQLiteCommand cmd = connect.CreateCommand();
            cmd.CommandText = @"SELECT  ExamMarks.ID, ExamMarks.ExamsID, ExamMarks.StudentsID, ExamMarks.Score,
                                Exams.Heading AS ExamName, Students.LastName AS StudentName
                                FROM ExamMarks LEFT JOIN Exams ON Exams.ID = ExamMarks.ExamsID
                                LEFT JOIN Students ON Students.ID = ExamMarks.StudentsID";
            var reading = cmd.ExecuteReader();
            while (reading.Read())
            {
                if (role != "Student" && string.IsNullOrWhiteSpace(Search))
                {
                    list.Add(new Marks
                    {
                        ID = Convert.ToInt32(reading["ID"]),
                        ExamName = reading["ExamName"].ToString(),
                        ExamID = Convert.ToInt32(reading["ExamsID"]),
                        StudentName = reading["StudentName"].ToString(),
                        StudentID = Convert.ToInt32(reading["StudentsID"]),
                        Score = reading["Score"].ToString()
                    });
                }
                else if(role != "Student" && (reading["ExamName"].ToString().Contains(Search)
                    || reading["StudentName"].ToString().Contains(Search)))
                {
                    list.Add(new Marks
                    {
                        ID = Convert.ToInt32(reading["ID"]),
                        ExamName = reading["ExamName"].ToString(),
                        ExamID = Convert.ToInt32(reading["ExamsID"]),
                        StudentName = reading["StudentName"].ToString(),
                        StudentID = Convert.ToInt32(reading["StudentsID"]),
                        Score = reading["Score"].ToString()
                    });
                }
                else if (reading["StudentsID"].ToString() == StudentID.ToString() && string.IsNullOrWhiteSpace(Search))
                {
                    list.Add(new Marks
                    {
                        ID = Convert.ToInt32(reading["ID"]),
                        ExamName = reading["ExamName"].ToString(),
```

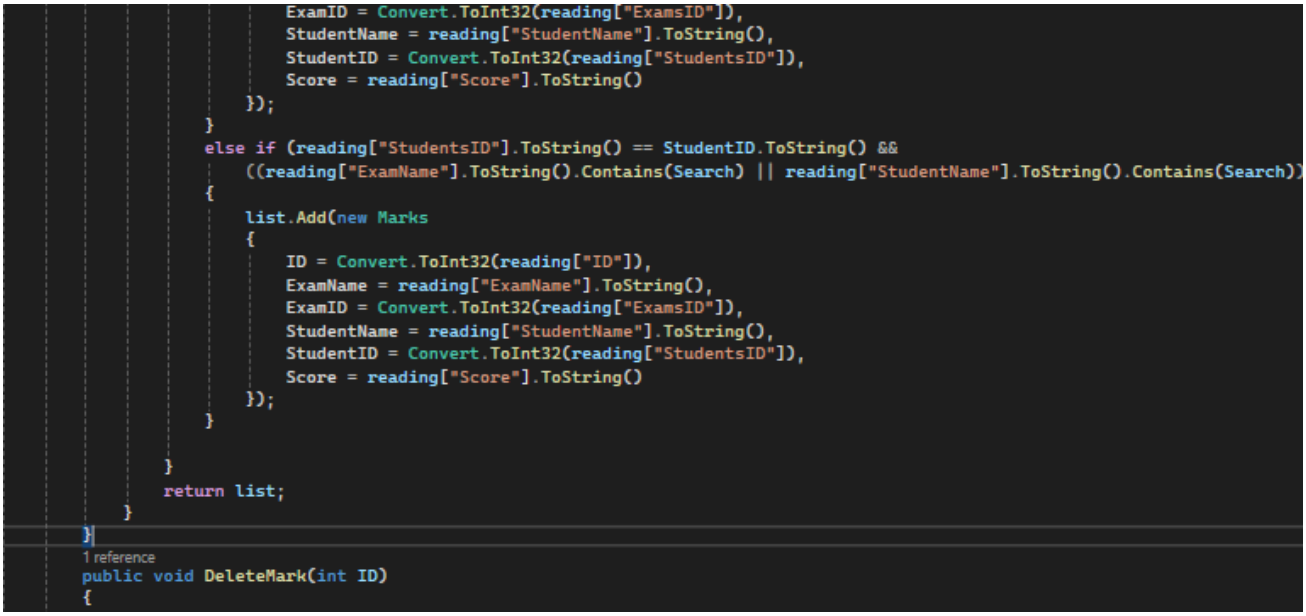


Figure 1MarkController-LaodMArks

Description

If the User Role is Student, Only his/her Marks Will be Displayed. Other User Role can See All Marks and also can Search.

- UserName Auto Assaign & Password Auto Create Based On Admin Selection

```
internal class UserController : CommanUses
{

    4 references
    public string SaveUser(Users user)
    {
        if (!string.IsNullOrWhiteSpace(user.UserName) && !string.IsNullOrWhiteSpace(user.Gmail) &&
            !string.IsNullOrWhiteSpace(user.CreatedDate) && !string.IsNullOrWhiteSpace(user.UserNameCreateType))
        {
             string GmailResult = GmailValidation(user.Gmail);
            if (GmailResult == "Invalid") { return "Failed"; }
            else
            {
                bool status =true;
                while (status)
                {
                    if (user.UserNameCreateType == "Manual")
                    {
                        using (UserCreation userCreation = new UserCreation())
                        {
                            if (userCreation.ShowDialog() == DialogResult.OK)
                            {
                                user.UserName = userCreation.Username;
                                user.Password = userCreation.Password;
                            }
                            else
                            {
                                MessageBox.Show("User Creation Cancelled!");
                                return "Failed";
                            }
                        }
                    }
                    if (user.Password == null) { user.Password = GeneratePassCode().ToString(); }
                    using (SQLiteConnection connection = DatabaseManager.GetConnection())
                    {
                        try
                        {
                            SQLiteCommand CMD = connection.CreateCommand();
                            CMD.CommandText = @"INSERT INTO Users(Name,GMail,Password,Role,CreatedDate,UpdatedDate)
                                VALUES(@name,@gmail,@password,@role,@createddate,@updateddate)";
                            CMD.Parameters.AddWithValue("@name", user.UserName);
                            CMD.Parameters.AddWithValue("@gmail", user.Gmail);
                            CMD.Parameters.AddWithValue("@password", user.Password);
                            CMD.Parameters.AddWithValue("@role", user.Role);
                            CMD.Parameters.AddWithValue("@createddate", user.CreatedDate);
                            CMD.Parameters.AddWithValue("@updateddate", user.UpdatedDate);
                            CMD.ExecuteNonQuery();
                            status = false;

                            return ($"User Created Successfully\nYour UserName is :{user.UserName}\nYour Password is :{user.Password}");
                        }
                        catch(SQLiteException ex) when (ex.ResultCode == SQLiteErrorCode.Constraint)
                        {
                            MessageBox.Show("This UserName Already Taken!");
                            using (UserCreation userCreation = new UserCreation(user.Password))
                            {
                                if (userCreation.ShowDialog() == DialogResult.OK)
                                {
                                    user.UserName = userCreation.Username;
                                    user.Password = userCreation.Password;
                                    continue;
                                }
                                else
                                {
                                    MessageBox.Show("User Creation Cancelled!");
                                    return "Failed";
                                }
                            }
                        }


                    }
                }

            }
            return "Failed";
        }
        else
        {
            MessageBox.Show("Please Fill All Details!"); return "Failed";
        }

    }
```

Figure 2UserController-auto Create passcode &UserName

Description

When Registering he can Set the Password to be auto and the Username to ne Lastname, or he can Choose Both Herself

- when click a textbox or Combo Box show error message on upper empty texboxes and comboboxes

```
6 references
internal class AdminController : CommanUses
{
    2 references
    public List<string> CheckEmptyVariables(Admins admin,string GMail)
    {
        List<string> Deta = new List<string>();
        foreach (PropertyInfo prop in admin.GetType().GetProperties())
        {
            if (prop.GetValue(admin)==null || string.IsNullOrWhiteSpace(prop.GetValue(admin).ToString()))
            {
                Deta.Add(prop.Name);
            }
        }
        if (string.IsNullOrWhiteSpace(GMail)) { Deta.Add("Gmail"); }
        return Deta;

    }
    1 reference
```

Figure 3AdminControlle-ValidationMethod

```
private void CheckEmptyFields(string CurrentPlace)
{
    List<string> Deta = new List<string>();
    Deta = adminControler.CheckEmptyVariables(admin,users.Gmail);
    if (Deta.Contains("FirstName")) { la_FirstName.Text = "*Enter Your FirstName"; }
    if (CurrentPlace == "LastName") {return; }
    if (Deta.Contains("LastName")) { la_LastName.Text = "*Enter Your LastName"; }
    if (CurrentPlace == "Gmail") { return; }
    if (Deta.Contains("Gmail")) { la_GMail.Text = "*Enter Your Gmail Address"; }
    if (CurrentPlace == "Phone") { return; }
    if (Deta.Contains("Phone")) { la_Phone.Text = "*Enter Your Mobile Number "; }
    if (CurrentPlace == "Address") { return; }
    if (Deta.Contains("Address")) { la_Address.Text = "Enter Your Address"; }
    if (CurrentPlace == "NicNo") { return; }
    if (Deta.Contains("NicNo")) { la_Nic.Text = "*Enter Your NIC Number"; }
    if(CurrentPlace == "radiomale" || CurrentPlace == "radiomale"){ return; }
    if (Deta.Contains("Gender")) { la_Gender.Text = "*Choose Your Gender"; }
}
```

Figure 4AdminRegisterForm- Show Error Method

```
1 reference
private void LastName_MouseClick(object sender, MouseEventArgs e)
{
    if (LastName.ForeColor != Color.Black) { LastName.Text = null; }
    CheckEmptyFields("LastName");
}
1 reference
private void Gmail_Click(object sender, EventArgs e)
{
    if(Gmail.ForeColor != Color.Black) { Gmail.Text = null; }
    CheckEmptyFields("Gmail");
}
1 reference
private void Phone_Click(object sender, EventArgs e)
{
    if(Phone.ForeColor!=Color.Black) { Phone.Text = null; }
    CheckEmptyFields("Phone");
}
1 reference
private void Address_Click(object sender, EventArgs e)
{
    if (Address.ForeColor != Color.Black) { Address.Text = null; }
    CheckEmptyFields("Address");
}
1 reference
private void NicNo_Click(object sender, EventArgs e)
{
    if (NicNo.ForeColor != Color.Black) {NicNo.Text= null; }
    CheckEmptyFields("NicNo");

}
1 reference
private void radiomale_Click(object sender, EventArgs e)
{
    CheckEmptyFields("radiomale");
```

Figure 5Admin RegisterForm-Calling ErrorMessage Mthod

- Update Method

```
        }
    }
    1 reference
    public void UpdateAdmin(Admins admin)
    {
        if (!string.IsNullOrWhiteSpace(admin.FirstName) && !string.IsNullOrWhiteSpace(admin.LastName) && !string.IsNullOrWhiteSpace(admin.NicNo) &&
            !string.IsNullOrWhiteSpace(admin.Phone) && !string.IsNullOrWhiteSpace(admin.Gender) && !string.IsNullOrWhiteSpace(admin.Address))
        {
            DialogResult result = MessageBox.Show("Make Changes for This Admin", "Confirmation", MessageBoxButtons.YesNo);
            if (result == DialogResult.Yes)
            {
                using (SQLiteConnection connect = DatabaseManager.GetConnection())
                {
                    using(SQLiteCommand cmd = connect.CreateCommand())
                    {
                        cmd.CommandText = @"UPDATE Admins SET FirstName=@fname, LastName=@lname, NicNumber=@nic, Phone=@phone, Gender=@gender,
                        Address=@address WHERE ID=@id";
                        cmd.Parameters.AddWithValue("@id", admin.Id);
                        cmd.Parameters.AddWithValue("@fname", admin.FirstName);
                        cmd.Parameters.AddWithValue("@lname", admin.LastName);
                        cmd.Parameters.AddWithValue("@nic", admin.NicNo);
                        cmd.Parameters.AddWithValue("@phone", admin.Phone);
                        cmd.Parameters.AddWithValue("@gender", admin.Gender);
                        cmd.Parameters.AddWithValue("@address", admin.Address);
                        cmd.ExecuteNonQuery();
                        MessageBox.Show("Updated Successfully");
                    }

                }
            }
        }
        else { MessageBox.Show("Fill All Details!"); }
    }
```

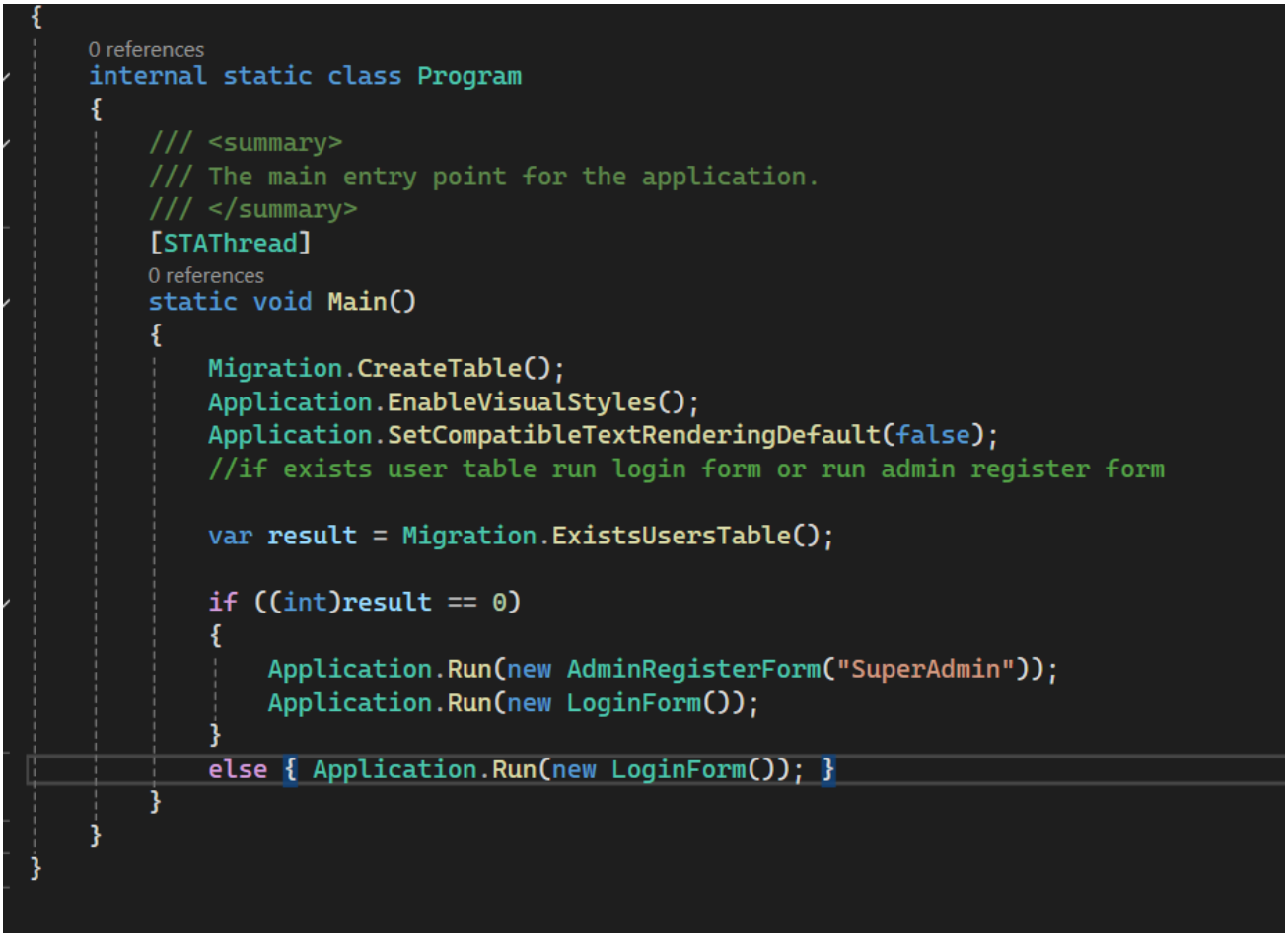
Figure 6AdminController-UpdateMethod

- When the application runs for the first time, display the registration form for the SuperAdmin, or else display the login form.

```
        }
        1 reference
        internal static object ExistsUsersTable()
        {
            using (SQLiteConnection connection =DatabaseManager.GetConnection())
            {
                SQLiteCommand cmd = connection.CreateCommand();
                cmd.CommandText = "SELECT Id FROM Users WHERE Id=@id";
                cmd.Parameters.AddWithValue("@id",1);
                int result = cmd.ExecuteNonQuery();
                return result;
            }
        }

    }
}
```

Figure 7Migration-Check UsersTable Empty



```
{
    0 references
    internal static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        0 references
        static void Main()
        {
            Migration.CreateTable();
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            //if exists user table run login form or run admin register form

            var result = Migration.ExistsUsersTable();

            if ((int)result == 0)
            {
                Application.Run(new AdminRegisterForm("SuperAdmin"));
                Application.Run(new LoginForm());
            }
            else { Application.Run(new LoginForm()); }
        }
    }
}
```

Figure 8Program.cs-

- Comman Validation Methods Are In ParentClass

```
5 references
internal class CommanUses
{
    1 reference
    public int GeneratePassCode()
    {
        Random random = new Random();
        int Passcode = random.Next(100000, 1000000);
        return Passcode;
    }

    4 references
    protected int GetLastInsertedId()
    {
        using (SQLiteConnection connect = DatabaseManager.GetConnection())
        {
            SQLiteCommand cmd = connect.CreateCommand();
            cmd.CommandText = "SELECT Id FROM Users ORDER BY Id DESC LIMIT 1;;";
            int lastId = Convert.ToInt32(cmd.ExecuteScalar());
            return lastId;
        }
    }
    4 references
    protected string PhoneValidation(string phone)
    {
        if (phone.Length != 10 || !phone.All(Char.IsDigit))
        {
            MessageBox.Show("Invalid MobileNo!");
            return "Invalid";
        }
    1 reference
    protected string GmailValidation(string email)
    {
        if (!email.EndsWith("@gmail.com", StringComparison.OrdinalIgnoreCase)
            || email.Length <= "@gmail.com".Length)
        {
            MessageBox.Show("Invalid Gmail Format!");
            return "Invalid";
        }
        else { return email; }
    }
    4 references
    protected string NicValidation(string nic)
    {
        if (nic.Length ==12 && nic.All(Char.IsDigit)) { return nic; }
        else if (nic.Length == 10 && nic.Substring(0, 9).All(Char.IsDigit) && nic.EndsWith("V")) { return nic; }
        else {MessageBox.Show("Invalid NicNo!"); return "Invalid"; }
    }
    2 references
    protected string CurrencyValidation(string currency)
    {
        try
        {
            double newcurrency = Convert.ToDouble(currency);
            return currency;
        }
        catch(Exception ex) { return "Invalid"; }
    }
}
```

Figure 9Comman Uses- Validation Methods

How the App Works:

- Login System: Authenticates users based on their role (SuperAdmin,Admin, Staff, Student, Lecturer).
- Role Dashboards: Different dashboards are shown depending on the user role. Data Operations: Add/Edit/Delete supported for Admin; restricted view-only for other roles.
- Rooms & Timetable: Admin allocates labs/halls while scheduling; others can only view.
- Exam & Marks: Staff and Lecturers can update marks; students can only view their own.